2.Write programs using the I/O system calls of UNIX/LINUX operating system(open,read,write,close, fcntl, seek, stat, opendir, readdir)

open,read,write, close

```
#include<sys/types.h>
#include<stdio.h>
#include <unistd.h>
#include <fcntl.h>
#define BUFSIZE
512intmain ()
{
int from, to, nr, nw, n;
charbuf[BUFSIZE],ch;
if((from=open("one.txt",O_RDONLY))<0)
{
printf("Error opening source file");
exit(1);
     }

     if ((to=creat("two.txt", O_RDWR)) < 0)


     {
     printf("Error creating destination file");
     exit(2);
     }
while((nr=read(from,buf,sizeof(buf))) !=0)
{
if(nr <0)
{
          printf("Errorreadingsourcefile");
          exit(3);

       }
       nw=0;
       do{
         if((n=write(to,&buf[nw],nr-nw))<0)
         {
         printf("Error writing destination file");
         exit(4);
         }
         nw+=n;
       }
       while (nw <nr);
     }
printf("successfullycopied thecontent fromfiel one.txtto two.txt");

     close(from);
```

```
    close(to);
}
```

INPUT
Create file name one.txt with some content OUTPUT
Successfully copied the content from file one.txt to two.txt
//you can see that program has created file two.txt and has content same as one.txt.

opendir,readdir

```
#include<stdio.h>
#include<dirent.h>
int main()
{
      Struct dirent* de;
      DIR *dr=opendir(".");
      if(dr==NULL)
      {
            printf("Could not open current Directory");
            return0;
      }
      while((de=readdir(dr))!=NULL)
                  printf("%s\n",de->d_name);
                  closedir(dr);

      return1;
      }
```

OUTPUT
// will be list of all the directories;
example output below:ashrc
.bash_history
.bash_logout
.fcfs.c.swp
.landscape
.motd_shown
.msgqs.c.swp
.msgsend.c.swp
.pro.c.swp
.profile
.sc.c.swp
.sudo_as_admin_successful
.viminfo
a.c
a.outd
irdir.c
file1fi
le1.cfi
le2file
2.c

5. Write C programs to illustrate the following IPC mechanisms
a)Pipes b)FIFOs c)MessageQueues d)SharedMemory

Pipes

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<unistd.h>
Int main()
{
	int fd1[2];
	intfd2[2];
	charfixed_str[]="Welcome";
	charinput_str[100];
	pid_t p;
	if(pipe(fd1)==-1)
	{
		fprintf(stderr,"pipe failed");
		return1;
	}
	if(pipe(fd2)==-1)
	{
	fprintf(stderr,"pipe failed");
	return1;
	}
	scanf("%s",input_str);
	p=fork();
	if(p<0){
		fprintf(stderr,"forkfailed");
		return1;
	}
	elseif(p>0){
		charconcat_str[100];
		close(fd1[0]);
		write(fd1[1],input_str,strlen(input_str)+1);
		close(fd1[1]);
		wait(NULL);
		close(fd2[1]);
		read(fd2[0],concat_str,100);
		printf("concatenated string %s\n",concat_str);
		close(fd2[0]);
	}
	else{
		close(fd1[1]);
		char concat_str[100];
		read(fd1[0],concat_str,100);
		intk=strlen(concat_str);
```

```
        int i;
        for(i=0;i<strlen(fixed_str);i++)
            concat_str[k++]=fixed_str[i];
            concat_str[k]='\0';
        close(fd1[0]);
        close(fd2[0]);
        write(fd2[1],concat_str,strlen(concat_str)+1);
        close(fd2[1]);
        exit(0);
    }
}
```

OUTPUT

```
ubuntu@DESKTOP-B8CV2UR:~$
./a.outHi
concatenated string
HiWelcomeubuntu@DESKTOP-
B8CV2UR:~$
```

FIFOs

Fifo writer code

```
#include<stdio.h>
#include<string.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<unistd.h>
Int main()
{
    int fd;
    char
    *myfifo="/home/ubuntu/myfifo";mkfi
    fo(myfifo,0666);
    chararr1[80],arr2[80];
    while(1)
    {
        fd=open(myfifo,O_WRONLY);
        fgets(arr2,80,stdin);
        write(fd, arr2,strlen(arr2)+1);
        close(fd);
        fd=open(myfifo,O_RDONLY);
        read(fd,arr1,sizeof(arr1));
        printf("USer2: %s\n",arr1);
        close(fd);
    }
    return0;
}
```

Fiforeaderscode

```
#include<stdio.h>
#include<string.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
	int fd;
	char *
	myfifo="/home/ubuntu/myfifo";mkfifo
	(myfifo,0666);
	char str1[80],str2[80];
	while(1)
	{
		fd=open(myfifo,O_RDONLY);
		read(fd,str1,80);
		printf("User1: %s\n",str1);
		close(fd);
		fd=open(myfifo,O_WRONLY);
		fgets(str2,80,stdin);
		write(fd,str2,strlen(str2)+1);
		close(fd);
	}
	return0;
}
```

OUTPUT

//simultaneouslyexecutebothwriteandreadercodeintwoterminals

6. Write C programs to simulate the following memory management techniques
a)Paging b)Segmentation

Paging

```
#include<stdio.h>
#include<conio.h>
intmain()
{
 intms,ps,nop,np,rempages,i,j,x,y,pa,offset;
 ints[10], fno[10][20];
printf("\nEnter the memory size -- ");
scanf("%d",&ms);
printf("\nEnterthepagesize--");
scanf("%d",&ps);
nop=ms/ps;
printf("\nThe no. of pages available in memory are -- %d ",nop);
printf("\nEnternumber of processes--");
 scanf("%d",&np);
rempages = nop;
for(i=1;i<=np;i++)
{
printf("\nEnterno.ofpagesrequired forp[%d]--",i);
 scanf("%d",&s[i]);
if(s[i]>rempages)
{
printf("\nMemory is Full");
break;
}
rempages=rempages-s[i];
printf("\nEnterpagetable forp[%d]---",i);
for(j=0;j<s[i];j++)
scanf("%d",&fno[i][j]);
}
printf("\nEnter Logical Address to find Physical Address ");
printf("\nEnter process no. and pagenumber and offset -- ");
scanf("%d %d %d",&x,&y, &offset);
if(x>np||y>=s[i]||offset>=ps)
printf("\nInvalid Process or Page Number or offset");
else
{
pa=fno[x][y]*ps+offset;
```

```
printf("\nThePhysicalAddressis--%d",pa);
}
getch();
}
```

OUTPUT

```
Enter the memory size – 1000 Enter the page size --
100Theno.ofpages available in memory are--10
Enternumberofprocesses--3
Enterno.ofpagesrequiredforp[1]--
4Enterpagetableforp[1]---8 6
9
5
Enter no. of pages required for p[2]--
5Enter pagetable for p[2] --- 1 4 5 7
3Enterno.ofpagesrequiredforp[3]—5


MemoryisFull
Enter Logical Address to find Physical Address Enter process no. and pagenumber and offset--2
3
60
ThePhysical Address is--760
```

Segmentation

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
structlist
{
int seg;
int base;
int limit;
struct list* next;
} *p;
void insert(structlist*q,intbase,intlimit,intseg)
{
if(p==NULL)
{
p=(structlist*)malloc(sizeof(structlist));
p->limit=limit;
p->base=base;
p->seg=seg;
```

7. Write C Programs to simulate page replacement policies
   a). FCFS
   b). LRU
   c). Optimal

AIM: To implement page replacement algorithms

ALGORITHM:

FCFS:
Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue

```
#include<stdio.h>
#include<conio.h>
int i,j,nof,nor,flag=0,ref[50],frm[50],pf=0,victim=-1;
void main()
{
clrscr();
printf("\n \t\t\t FIFI PAGE REPLACEMENT ALGORITHM");
printf("\n Enter no.of frames....");
 scanf("%d",&nof);
printf("Enter number of reference string..\n");
 scanf("%d",&nor);
printf("\n Enter the reference string..");
 for(i=0;i<nor;i++)
 scanf("%d",&ref[i]);
printf("\nThe given reference string:");
 for(i=0;i<nor;i++)
 printf("%4d",ref[i]);
 for(i=1;i<=nof;i++)
 frm[i]=-1;
 printf("\n");
 for(i=0;i<nor;i++)
 {
   flag=0;
printf("\n\t Reference np%d->\t",ref[i]);
   for(j=0;j<nof;j++)
   {
     if(frm[j]==ref[i])
     {
       flag=1;
       break;
     }}
      if(flag==0)
   {
     pf++;
```

```
 victim++;
    victim=victim%nof;
    frm[victim]=ref[i];

    for(j=0;j<nof;j++)
    printf("%4d",frm[j]);
  }
 }
printf("\n\n\t\t No.of pages faults...%d",pf);
getch();
}
```

OUTPUT:

FIFO PAGE REPLACEMENT ALGORITHM

```
Enter no.of frames....4
Enter number of reference string..
6

 Enter the reference string..
5 6 4 1 2 3



The given reference string:
      .....   5   6   4   1   2   3

       Reference np5->5  -1  -1  -1
       Reference np6->         5   6  -1  -1
       Reference np4->         5   6   4  -1
       Reference np1->         5   6   4   1
       Reference np2->         2   6   4   1
       Reference np3->         2   3   4   1

No.of pages faults...6
```